Non-Colluding Attacks Identification in Distributed Computing

Arnav Solanki, Martina Cardone, Soheil Mohajer
University of Minnesota, Minneapolis, MN 55404, USA
Email: {solan053, cardo089, soheil}@umn.edu

***Abstract*—This paper studies a distributed computing setting in which the computing task consists of multiplying a matrix by a vector. A number of worker machines are attacked, i.e., the result of their computation is maliciously perturbed by some adversaries. In particular, the focus is on the case where these adversaries are *non-colluding* and *non-communicating* and hence they cannot jointly perturb the results of all the attacked worker machines. First, a condition that ensures that the result of the computing task can be successfully recovered with high probability is derived as a function of the setting parameters. Then, a probabilistic mechanism inspired by group testing is proposed to identify the set of the attacked worker machines, and the corresponding probability of error is derived.**

I. Introduction

Today, datasets have reached overwhelming sizes and complexities, which make them difficult to be processed using traditional approaches. Distributed computing embraces the idea of parallelizing the computations to enable large-scale data processing. The effort of a computing task over these massive and complex datasets can indeed be distributed by assigning sub-tasks across a network of worker machines that operate in parallel. A critical aspect in the design of modern distributed computing systems concerns the *security* of the computation process. With the ever-growing computational power at our disposal, these systems are indeed susceptible to different types of message tampering attacks, in which an adversary (e.g., virus) might maliciously perturb the results computed by some worker machines.

In this paper, we consider a distributed computing setting in which the task consists of multiplying a matrix by a vector. We assume that some of the worker machines are *attacked*, i.e., the result of their computation is maliciously perturbed. In particular, we focus on a scenario where the worker machines are geographically distributed, and belong to different systems. For such a scenario, it is reasonable to assume that adversaries attacking different worker machines are non-colluding/non-communicating. As such, they cannot jointly perturb the results of all the attacked worker machines.

This scenario is different from the setting recently analyzed in [1], which assumes attackers that collaborate in perturbing the results computed by several worker machines. The assumption of non-colluding/non-communicating adversaries finds applicability in the context of online crowdsourcing [2], [3], where it can be difficult to ensure high-quality and reliable results from the entire large pool of Internet users.

M. Cardone was supported in part by the U.S. National Science Foundation under Grant CCF-1849757.

We first derive a condition under which, with the assumed data perturbation and employed technique of distributing the data, it is guaranteed that the master node can successfully recover the result of the computing task with high probability. This condition is $k < n-\ell$, where n is the number of worker machines, ℓ is the number of attacked worker machines, and k is the number of data splits. Then, we propose a probabilistic mechanism to identify the set of the ℓ attacked worker machines, and we compute the corresponding probability of error. Based on this knowledge, the master node can decide how to distribute the computing efforts in the future.

Related Work. In this work, data is distributed across the worker machines by adopting a *coding* strategy, which aims at offering a suitable data redundancy mechanism robust to message tampering. In the context of distributed computing, coded solutions have been shown to provide several benefits. For instance, coding has been used to: (i) alleviate the impact of stragglers, i.e., worker machines that are significantly slow [4], [5], [6]; (ii) reduce the cost of communication and bandwidth usage [6], [7], [8]; and (iii) ensure that data remains confidential from the worker machines [9], [1], and is resilient against attacks from collaborative adversaries [1].

To identify the set of the attacked worker machines, we propose a probabilistic mechanism inspired by group testing. Group testing was introduced in [10] and aims at providing mechanisms to efficiently (i.e., with the least number of tests) identify a set of *defective* items in a large pool of items. In particular, each test consists of a group of items, and the answer is positive if the group includes at least one defective item (otherwise the answer is negative). Several classifications exist for group testing (e.g., adaptive or non-adaptive, probabilistic or combinatorial); we refer an interested reader to [11] for a comprehensive overview.

Paper Organization. Section II describes the distributed computing setting, and formulates the problem. Section III provides a condition that ensures that the result of the computing task can be successfully recovered with high probability. Section IV designs a probabilistic mechanism to identify the attacked worker machines, and computes the corresponding error probability.

II. Distributed Computing Setup

We consider a canonical distributed computing scenario, which consists of a master node and n non-communicating worker machines. Fig. 1 provides a representation of the distributed computing setting of interest.

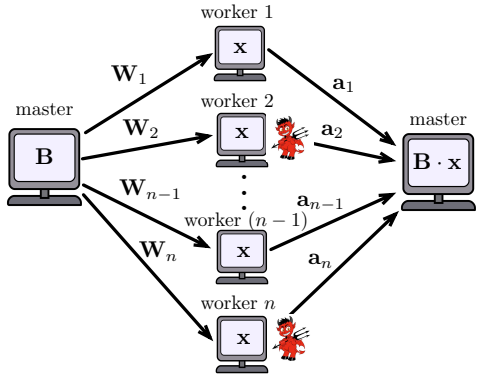

Fig. 1: Distributed computing setting in which there are n worker machines, out of which ℓ are attacked. $\mathbf{W}_i$ and $\mathbf{a}_i$ with $i \in [1:n]$ are defined in (1) and (2), respectively.

The master node has a matrix $\mathbf{B} \in \mathbb{F}_q^{r\times c}$, and it seeks to retrieve the product $\mathbf{B}\cdot\mathbf{x}$, where $\mathbf{x} \in \mathbb{F}_q^{c\times 1}$ is stored at each of the n worker machines[1]. We assume that $r \gg 1$, and hence, in order to speed the process, the effort of the computing task (i.e., matrix multiplication) can be distributed across the n worker machines. Each worker machine has a limited computational capability, which is captured by the parameter $r' \leq r$. In particular, r' represents the number of rows of $\mathbf{B}$ that each worker machine can receive and process.

In this setting, among the n worker machines, a subset of cardinality ℓ of them are attacked by some adversaries (e.g., viruses). It is assumed that the identity of the ℓ attacked worker machines is not known by the master node. In particular, these ℓ adversaries are active, i.e., they maliciously perturb the result of the computing task with some additive noise. We assume that these ℓ adversaries are non-colluding/non-communicating, and that noises introduced by different attackers are *independent*.

In order to leverage the worker machines, while mitigating the effect of those that are attacked, the master node employs a set of encoding functions $f_i(\cdot) : \mathbb{F}_q^{r\times c} \to \mathbb{F}_q^{r'\times c}$ for $i \in [1:n]$, to encode and distribute data across the n worker machines. In particular, the i-th worker machine will receive

$$\mathbf{W}_i = f_i(\mathbf{B}). \tag{1}$$

At this point, each worker machine will compute the product $\mathbf{W}_i \cdot \mathbf{x}$, and return this to the master node. If a worker machine is attacked, then its message sent to the master node is perturbed and different from $\mathbf{W}_i \cdot \mathbf{x}$. We model this perturbation by an additive noise. Denoting by $\mathcal{L}$ the set of the attacked worker machines, the i-th worker machine sends

$$\mathbf{a}_i = \mathbf{W}_i \cdot \mathbf{x} + \mathbb{1}_{\mathcal{L}}(i)\mathbf{Z}_i, \tag{2}$$

to the master node, where: (i) $\mathbb{1}_{\mathcal{L}}(i)$ is the indicator function, i.e., it is 1 if $i \in \mathcal{L}$ and 0 otherwise, and (ii) $\mathbf{Z}_i \in \mathbb{F}_q^{r'\times 1}$ is the noise vector introduced by worker machine $i \in [1:n]$. In particular, we assume that the *components* of $\mathbf{Z}_i, i \in [1:n]$, are drawn from a uniform distribution, and can be arbitrarily correlated. However, since the adversaries are not communicating, we assume that the vectors $\mathbf{Z}_i$ and $\mathbf{Z}_j$ are independent for all $(i,j) \in [1:n]^2$ and $i \neq j$.

[1]Operations take place over finite field of dimension q.

The master node, once it receives the vectors $\mathbf{a}_i$, $\forall i \in [1:n]$, will apply the decoding function $g(\cdot) : \left(\mathbb{F}_q^{r'\times 1}\right)^n \to \mathbb{F}_q^{r\times 1}$ on them with the goal to retrieve the product $\mathbf{B}\cdot\mathbf{x}$.

The distributed setting described above is therefore parameterized by the tuple (n,ℓ,r,r'). We say that this tuple is *achievable* if there exist encoding and decoding functions $f_i(\cdot), i \in [1:n]$, and $g(\cdot)$ such that

$$g(\mathbf{a}_1,\ldots,\mathbf{a}_n) = \mathbf{B}\cdot\mathbf{x}. \tag{3}$$

In the next section, we provide a condition on (n,ℓ,r,r') that indeed ensures that the master node can correctly retrieve $\mathbf{B}\cdot\mathbf{x}$. Towards this end, we propose a linear design for the encoding and decoding functions $f_i(\cdot), i \in [1:n]$, and $g(\cdot)$.

III. Condition for Correctness of the Distributed Computing Task

In this section, we derive a sufficient condition on the tuple (n,ℓ,r,r'), which ensures that the master node can correctly retrieve the product $\mathbf{B}\cdot\mathbf{x}$. In particular, our main result is stated in the theorem below.

Theorem 1. *Consider a distributed computing scenario with n non-colluding worker machines, out of which ℓ are attacked. Let $\mathbf{B} = \left[\mathbf{B}_1^T \quad \mathbf{B}_2^T \quad \ldots \quad \mathbf{B}_k^T\right]^T$, where $k = \lceil r/r' \rceil$, and each $\mathbf{B}_i, i \in [1:k]$, has r' rows[2]. Then, whenever $k < n-\ell$, the master node can correctly retrieve the product $\mathbf{B}\cdot\mathbf{x}$ with high probability over a large enough finite field.*

Before delving into the proof of Theorem 1, we state the following remark, which provides a similar condition as in Theorem 1 under the assumption of collaborative attackers.

Remark 1. *Consider a distributed computing scenario as in Theorem 1, but with collaborative attackers. Then, the master node can correctly retrieve the product $\mathbf{B}\cdot\mathbf{x}$ whenever $k < n-2\ell$ [1]. Thus, it follows that, if we consider k and n as fixed, then with non-colluding attackers, we can tolerate twice the number of attacked worker machines than in the case of collaborative attackers.*

The remainder of this section is dedicated to the proof of Theorem 1. In particular, we next propose a design of the encoding and decoding functions $f_i(\cdot), i \in [1:n]$, and $g(\cdot)$ in (1) and (3) which ensure that, whenever $k < n-\ell$, then the master node can retrieve the product $\mathbf{B}\cdot\mathbf{x}$ with high probability over a large enough finite field.

Encoding Functions. We define

$$\mathbf{W}_i = f_i(\mathbf{B}) = \sum_{j=1}^{k} \alpha_i^{j-1}\mathbf{B}_j, \tag{4}$$

where α_i's are the distinct coefficients of the tall Vandermonde matrix $\mathbf{V} \in \mathbb{F}_q^{n\times k}$ of the form

$$\mathbf{V} = \begin{bmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \ldots & \alpha_n^{k-1} \end{bmatrix}. \tag{5}$$

[2]Note that, if $kr' > r$, then each of the last $kr'-r$ rows of $\mathbf{B}_k$ will be a zero vector of dimension $1\times c$.

Decoding Function. The master node, once it receives the n vectors $\mathbf{a}_i$'s from the worker machines, it selects a subset of $k+1$ of them. Since ℓ worker machines are attacked, the number of *good* choices (i.e., choices that do not contain any attacked worker machine) is $\binom{n-\ell}{k+1}$. Thus, we need $k \leq n-\ell-1$. We now show that, under such condition, the master node can successfully retrieve the product $\mathbf{B}\cdot\mathbf{x}$ with high probability over a large enough finite field. We let $\mathcal{U}$ be the ordered set that contains the $k+1$ worker machines that the master node has selected. Note that $\mathcal{U} \subset [1:n]$, and $|\mathcal{U}| = k+1$. We also define $\tilde{\mathbf{V}} \in \mathbb{F}_q^{|\mathcal{U}|\times|\mathcal{U}|}$ as a Vandermonde matrix of a form similar to the one in (5), where we use the coefficients $\alpha_i, \forall i \in \mathcal{U}$. Since $\tilde{\mathbf{V}}$ is a Vandermonde matrix, it is invertible and the $|\mathcal{U}|$-th row of its inverse – denoted as $\mathbf{r}^{\mathcal{U}}$ – has components from $\mathbb{F}_q$ of the form [12]

$$r_i^{\mathcal{U}} = \frac{1}{\prod_{\substack{m\in\mathcal{U}\\ m\neq i}} (\alpha_i - \alpha_m)}, \tag{6}$$

for all $i \in \mathcal{U}$. Note that all the components of $\mathbf{r}^{\mathcal{U}}$ are different from zero. With this definition, it therefore follows that

$$\mathbf{r}^{\mathcal{U}} \cdot \tilde{\mathbf{V}} = \mathbf{e}_{|\mathcal{U}|},$$

where $\mathbf{e}_{|\mathcal{U}|}$ is a row vector of all zeros except a 1 in position $|\mathcal{U}|$. Consider the submatrix of $\tilde{\mathbf{V}}$ where only the columns in $[1:k]$ are retained. This matrix is also a submatrix of $\mathbf{V}$ in (5), where only the rows indexed by $\mathcal{U}$ are retained. We let $\mathbf{V}_{\mathcal{U}}$ denote this matrix and, since $|\mathcal{U}| = k+1$, we obtain

$$\sum_{i\in\mathcal{U}} r_i^{\mathcal{U}} \alpha_i^{j-1} = 0, \ \forall j \in [1:k]. \tag{7}$$

Now, the master node, by using (6), performs the following operation

$$\begin{aligned}
\mathbf{d}^{\mathcal{U}} &= \sum_{i\in\mathcal{U}} r_i^{\mathcal{U}} \cdot \mathbf{a}_i \\
&\overset{(a)}{=} \sum_{i\in\mathcal{U}} r_i^{\mathcal{U}} \left(\mathbf{W}_i \cdot \mathbf{x} + \mathbb{1}_{\mathcal{L}}(i)\mathbf{Z}_i\right) \\
&\overset{(b)}{=} \sum_{i\in\mathcal{U}} r_i^{\mathcal{U}} \left[\left(\sum_{j=1}^{k} \alpha_i^{j-1}\mathbf{B}_j\right)\cdot\mathbf{x} + \mathbb{1}_{\mathcal{L}}(i)\mathbf{Z}_i\right] \\
&= \sum_{j=1}^{k}\sum_{i\in\mathcal{U}} \left(r_i^{\mathcal{U}}\alpha_i^{j-1}\right)\mathbf{B}_j\cdot\mathbf{x} + \sum_{i\in\mathcal{U}} r_i^{\mathcal{U}}\mathbb{1}_{\mathcal{L}}(i)\mathbf{Z}_i \\
&\overset{(c)}{=} \sum_{i\in\mathcal{U}} r_i^{\mathcal{U}}\mathbb{1}_{\mathcal{L}}(i)\mathbf{Z}_i,
\end{aligned} \tag{8}$$

where the equalities above follow since: (a) by using the expression in (2); (b) by using the expression in (4); (c) by using the expression in (7).

Since all the r_i's in (6), with $i \in \mathcal{U}$, are different from zero, we readily obtain that if $\mathcal{U}\cap\mathcal{L} = \varnothing$, then $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$. However, we also note that $\mathbf{d}^{\mathcal{U}}$ can still be equal to the zero vector even when $\mathcal{U}\cap\mathcal{L} \neq \varnothing$. Under our assumption on the noise vectors and their components in Section II, and over a finite field of dimension q, the probability of this event to occur is at most equal to $1/q$. Thus, for a large enough field size (i.e., q), this probability vanishes.

Once the master node selects a set of $k+1$ worker machines for which $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$. in (8), then it can also correctly retrieve (with probability equal to $1-1/q$) the product $\mathbf{B}\cdot\mathbf{x}$ by using the following procedure. First, the master node selects any k worker machines from the set $\mathcal{S}\subset\mathcal{U}$, with $|\mathcal{S}|=k$, and stacks together the corresponding k vectors $\mathbf{a}_i$'s. With this, the master node obtains

$$\mathbf{A}_{\mathcal{S}} = \left[\mathbf{a}_i^T : i \in \mathcal{S}\right]^T = \left(\mathbf{V}_{\mathcal{S}} \otimes \mathbf{I}_{r'}\right)\mathbf{B}\cdot\mathbf{x}, \tag{9}$$

where: (i) $\mathbf{A} = \begin{bmatrix}\mathbf{a}_1^T & \mathbf{a}_2^T & \dots & \mathbf{a}_n^T\end{bmatrix}^T$ and $\mathbf{A}_{\mathcal{S}}$ is the subvector of $\mathbf{A}$, where only the $\mathbf{a}_i$'s such that $i \in \mathcal{S}$ are retained; (ii) $\mathbf{V}_{\mathcal{S}}$ is the submatrix of $\mathbf{V}$ in (5) where only the rows indexed by $\mathcal{S}$ are retained; and (iii) $\otimes$ is the Kronecker product. Now, the master node premultiplies $\mathbf{A}_{\mathcal{S}}$ in (9) by $\left(\mathbf{V}_{\mathcal{S}}\otimes\mathbf{I}_{r'}\right)^{-1} = \mathbf{V}_{\mathcal{S}}^{-1}\otimes\mathbf{I}_{r'}$, and gets

$$g(\mathbf{a}_1,\dots,\mathbf{a}_n) = \left(\mathbf{V}_{\mathcal{S}}^{-1}\otimes\mathbf{I}_{r'}\right)\mathbf{A}_{\mathcal{S}} = \mathbf{B}\cdot\mathbf{x}. \tag{10}$$

This concludes the proof of Theorem 1.

In the analysis above, we have shown that if $k < n-\ell$, then the master node can recover the product $\mathbf{B}\cdot\mathbf{x}$ correctly with high probability. Now, a natural question that arises is: How many operations does the master node need to perform to recover $\mathbf{B}\cdot\mathbf{x}$ correctly (with high probability)? The next lemma answers this question on an average sense, when at each round the master node uniformly at random selects $k+1$ indices out of the n available ones and computes (8).

Lemma 2. *Assume that, at each round, the master node uniformly at random selects $k+1$ indices out of the n available ones and computes* (8)*. Then, in order to find a set of $k+1$ worker machines for which $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$ in* (8)*, the average number of rounds needed is $\frac{\binom{n}{k+1}}{\binom{n-\ell}{k+1}}$.*

The result in Lemma 2 follows since, under the assumption of uniform random selection of the $k+1$ indices, the random variable R representing the number of rounds needed by the master node to obtain $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$ in (8) is distributed according to a geometric distribution with parameter p, i.e., $P[R=r] = (1-p)^{r-1}p$, where $p = \frac{\binom{n-\ell}{k+1}}{\binom{n}{k+1}}$ for which $\mathbb{E}[R] = \frac{1}{p}$.

IV. Group Testing Approach to Identify the Attacked Worker Machines

In this section, we seek to design an algorithm to identify which are the ℓ worker machines that are attacked out of the n available ones. Towards this end, we propose a probabilistic approach inspired by group testing, whose performance is provided in the following theorem.

Theorem 3. *There exists a probabilistic testing mechanism that allows to identify the ℓ attacked worker machines with a probability of error* $\Pr\{\mathrm{E}\}$ *such that*

$$\Pr\{\mathrm{E}\} \leq n(1-\pi^\star)^M + \frac{1}{q},$$

where M is the number of tests and

$$\pi^\star = \max_{k+1\leq t\leq n-\ell} \frac{\binom{n-\ell-1}{t-1}}{\binom{n}{t}}.$$

The remainder of this section is dedicated to the proof of Theorem 3. In particular, in what follows, we start by providing some definitions from the group testing literature that will be used for the proof of Theorem 3.

A. Group Testing Problem

In the classical group testing problem, the goal is to construct a set of *tests* that allow to identify the defective items while minimizing the number of tests M needed. In particular, each test consists of a group of items, and the answer is positive if the group includes at least one defective item. By using our notation, we let n be the total number of items, and ℓ be the number of defective items. The group testing problem can be mathematically formulated as follows

$$\mathbf{y} = \mathbf{M} \odot \mathbf{u}, \tag{11}$$

where $\odot$ indicates that the arithmetic is Boolean, and where: (i) $\mathbf{y} \in \mathbb{F}_2^{M\times 1}$ is the test vector, i.e., $y_i = 0$ if the i-th test is negative, and $y_i = 1$ if the i-th test is positive (y_i indicates the i-th component of $\mathbf{y}$, with $i \in [1 : M]$); (ii) $\mathbf{M} \in \mathbb{F}_2^{M\times n}$ is the so-called *contact* matrix with $M_{ij} = 1$ if test $i \in [1 : M]$ contains item $j \in [1 : n]$, and $M_{ij} = 0$ otherwise; (iii) $\mathbf{u} \in \mathbb{F}_2^{n\times 1}$ is the vector that indicates the defective items among the population, i.e., $u_i = 1$ if item $i \in \mathcal{L}$ and $u_i = 0$ otherwise; note that $\mathbf{u}$ is an ℓ-sparse vector.

The goal of group testing is to design the contact matrix $\mathbf{M}$ in (11) such that M is as small as possible, and the vector $\mathbf{u}$ can be successfully identified. In group testing, a well-studied class of contact matrices is the one of *disjunct* matrices [11]. We next provide the formal definition of such matrices.

Definition 1. *Let $\mathbf{M} = \begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 \dots \mathbf{m}_n\end{bmatrix}$ be a Boolean matrix where $\mathbf{m}_i \in \mathbb{F}_2^{M\times 1}, i \in [1 : n]$. The matrix $\mathbf{M}$ is called ℓ-disjunct if, for every column $\mathbf{m}_i$ and every choice of ℓ columns different from $\mathbf{m}_i$, there is at least one row at which the entry corresponding to $\mathbf{m}_i$ is 1 and those corresponding to the other selected ℓ columns are all zeros.*

An appealing property of disjunct matrices is that they allow to distinguish sparse Boolean vectors, as stated in the following proposition [11].

Proposition 4. *Assume that an ℓ-disjunct matrix $\mathbf{M}$ with n columns is used as the contact matrix. Then, the test outcomes obtained by using the scheme on two distinct ℓ-sparse vectors of length n must differ in at least one element. In other words, if $\mathbf{u}_1$ and $\mathbf{u}_2$ are two distinct ℓ-sparse vectors, then $\mathbf{y}_1 = \mathbf{M} \odot \mathbf{u}_1$ and $\mathbf{y}_2 = \mathbf{M} \odot \mathbf{u}_2$ differ in at least one element.*

Although our problem of identifying the ℓ attacked worker machines has similarities with group testing, we next highlight two fundamental differences. These two differences point out that results derived in group testing can not be readily applied to our distributed computing setting.

Difference 1. In group testing $M = O(n)$ since, in a naive approach, each item in the pool can be tested individually. In this case, after $O(n)$ tests, all the ℓ defective items will be identified. The optimum solution of the group testing problem consists of optimizing the size of the pool (as a function of n and ℓ), in order to minimize M. In our scenario, testing each of the n worker machines separately is not helpful. This is because, in order to retrieve one possible result of the computing task (i.e., $\mathbf{B} \cdot \mathbf{x}$), the master node needs to collect together the results returned by k worker machines. In other words, the size of the pool in our scenario depends on k, and hence we cannot necessarily use the optimum contact matrix. Thus, in our scenario we might need a number of tests M that is larger than n to identify the ℓ attacked worker machines.

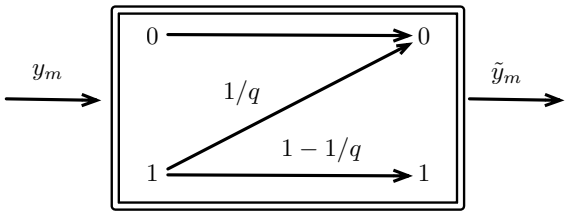

Fig. 2: Z-channel.

Difference 2. In group testing, the probability of error associated to a given number of tests M is proven to vanish as the number of items n increases. Differently, in our scenario the number of worker machines n is a fixed parameter and the only growing parameter that can offer a vanishing probability of error is q, i.e., the size of the finite field.

B. Mapping to Group Testing

We now show that our distributed computing setting can be studied within the framework of group testing. Towards this end, we leverage the proof of Theorem 1. Specifically, we use the fact that the master node selects a set $\mathcal{U} \subseteq [1 : n]$. However, different from the proof of Theorem 1, we now allow $|\mathcal{U}| > k + 1$ (note also $|\mathcal{U}| \leq n$). In other words, $\mathcal{U}$ becomes the pool of the group testing framework, and hence we would like to obtain $|\mathcal{U}|$ as small as possible (i.e., we seek to minimize the number of tests that we need for identifying the attacked worker machines).

Recall that we can obtain $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$ in (8) in two cases: (i) with probability one when $\mathcal{U} \cap \mathcal{L} = \varnothing$, and (ii) with probability $1/q$ when $\mathcal{U} \cap \mathcal{L} \neq \varnothing$. This second case, by using terminology from the group testing literature [13], corresponds to a scenario where our test (i.e., computation of $\mathbf{d}^{\mathcal{U}}$) is negative (i.e., $\mathbf{d}^{\mathcal{U}} = \mathbf{0}_{r'\times 1}$) even if our pool (i.e., $\mathcal{U}$) contains one or more defective items (i.e., attacked worker machines). This can be modeled as follows. Let $\mathbf{y} \in \mathbb{F}_2^{M\times 1}$ be the test vector of the classical group testing problem, where each test is assumed to be performed over a pool $\mathcal{U}$ as described above (i.e., such that $k + 1 \leq |\mathcal{U}| \leq n$). In particular, we let $\mathcal{U}_m$ be the pool associated with the m-th component of $\mathbf{y}$. Now each component of $\mathbf{y}$ is passed through the Z-channel in Fig. 2, whose output is denoted by $\tilde{\mathbf{y}}$. In other words, for all $m \in [1 : M]$, we have:

1) If $y_m = 0$, then $\tilde{y}_m = 0$ with probability one. This corresponds to having $\mathbf{d}^{\mathcal{U}_m} = \mathbf{0}_{r'\times 1}$ since $\mathcal{U}_m \cap \mathcal{L} = \varnothing$.
2) If $y_m = 1$, then $\tilde{y}_m = 0$ with probability $1/q$, and $\tilde{y}_m = 1$ with probability $1 - 1/q$. Note that the case $\tilde{y}_m = 0$ corresponds to having $\mathbf{d}^{\mathcal{U}_m} = \mathbf{0}_{r'\times 1}$ even if $\mathcal{U}_m \cap \mathcal{L} \neq \varnothing$.

Example. Let us consider a distributed computing system with $n = 20$ worker machines, out of which at most $\ell = 2$ are attacked. The parameters r and r' are such that $\mathbf{B}$ is split

into $k = 3$ parts, namely $\mathbf{B} = \begin{bmatrix}\mathbf{B}_1^T & \mathbf{B}_2^T & \mathbf{B}_3^T\end{bmatrix}^T$. Note that we have $k = 3 \leq 17 = n - \ell - 1$. For encoding, we use a 20×3 Vandermonde matrix $\mathbf{V}_{i,j} = i^{j-1}$, with $i \in [1:n]$ and $j \in [1:k]$. For example, the matrix sent to the forth worker machine is given by $\mathbf{W}_4 = \mathbf{B}_1 + 4\mathbf{B_2} + 16\mathbf{B}_3$ (operations are performed modulus q, which is a large integer number).

Our group testing approach provides us with a contact matrix of size $M \times n$. In particular, we let the rows of $\mathbf{M}$ be chosen uniformly among all the binary vectors of length $n = 20$ and Hamming weight $t = 6$. Our choice is motivated by the fact that the optimum value of t in Theorem 3 (i.e., the one that maximizes $\pi^\star$) is $t = 6$, which leads to $\pi^* \approx 0.16$. This choice will become clear in Section IV-C. Now, let us consider two rows of such $\mathbf{M}$ given by

$$\mathbf{M} = \begin{bmatrix} 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \\ \vdots \end{bmatrix}.$$

The first row corresponds to $\mathcal{U}_1 = \{2, 5, 6, 11, 13, 18\}$, and the second row corresponds to $\mathcal{U}_2 = \{6, 10, 12, 13, 18, 20\}$. We have to compute the parity defined in (8) for each group. For the sake of illustration, let us assume $\mathcal{L} = \{12, 20\}$. Since $\mathcal{U}_1 \cap \mathcal{L} = \varnothing$, we have $\mathbf{d}^{\mathcal{U}_1} = \mathbf{0}_{r' \times 1}$. However, for the second group we have $\mathcal{U}_2 \cap \mathcal{L} = \{12, 20\}$. For this group we have $\mathbf{d}^{\mathcal{U}_2} = r_{12}^{\mathcal{U}_2}\mathbf{Z}_{12} + r_{20}^{\mathcal{U}_2}\mathbf{Z}_{20}$, where $r_{12}^{\mathcal{U}_2} \neq 0$ and $r_{20}^{\mathcal{U}_2} \neq 0$. Therefore, $\mathbf{d}^{\mathcal{U}_2} = \mathbf{0}_{r' \times 1}$ if and only if the noises introduced by the two attacked worker machines satisfy a parity equation. This, since the noises are generated independently, occurs with a probability of at most $1/q$. Hence, with high probability, the test vector will be $\mathbf{y} = \begin{bmatrix}0 & 1 & \cdots\end{bmatrix}^T$.

By having more rows of the contact matrix $\mathbf{M}$ and the test vector $\mathbf{y}$, we can apply the group testing algorithm to identify the attacked worker machines. ■

C. Construction of the Contact Matrix

We now consider a probabilistic construction of the contact matrix $\mathbf{M}$, and show that it allows to identify the ℓ attacked worker machines with error probability given in Theorem 3. Each row of $\mathbf{M}$ is chosen uniformly from the set $\mathcal{T}$ that contains all vectors of length n and Hamming weight of $t \geq k+1$. In other words, each of the M rows of $\mathbf{M}$ corresponds to a possible choice of the set $\mathcal{U}$ in Section IV-B (recall $k + 1 \leq |\mathcal{U}| \leq n$). In particular, for the m-th row of $\mathbf{M}$ the corresponding $\mathcal{U}_m$ is the set that contains all indexes $i \in [1 : n]$ such that $M_{mi} = 1$ (see also the example in Section IV-B).

With such a probabilistic construction of $\mathbf{M}$, we might have two events E_1 and E_2 that may prevent the correct identification of the ℓ attacked worker machines:

- E_1: The contact matrix $\mathbf{M}$ is not ℓ-disjunct. In fact, note that Proposition 4 ensures that, if $\mathbf{M}$ is ℓ-disjunct, then it is possible to successfully recover the vector $\mathbf{u}$;
- E_2: There exist $\mathbf{y}_1$ and $\mathbf{y}_2$ that, after being passed through the Z-channel in Fig. 2, generate the same $\tilde{\mathbf{y}}$.

We start by analyzing the probability associated to E_1, which is referred to as $\Pr\{\mathrm{E}_1\}$. Consider the set $\mathcal{L}$ of ℓ columns of $\mathbf{M}$, and any column index i so that $i \notin \mathcal{L}$. Then, according to Definition 1, we say that a row of $\mathbf{M}$ is *good* for $\mathcal{L}$ and the choice of i if, at that row, the i-th column of $\mathbf{M}$ has a 1 and all the columns in $\mathcal{L} \setminus \{i\}$ have zeros. Thus, the probability that a row that belongs to $\mathcal{T}$ is good is

$$\Pr\{\text{row is good}\} = \frac{\binom{n-\ell-1}{t-1}}{\binom{n}{t}} = \pi. \tag{12}$$

Note that we need $t \leq n - \ell$. Thus, we have a failure probability if there are no good rows. Since each row is selected uniformly at random from $\mathcal{T}$, then all the events "row is good" in (12) are independent. It therefore follows that the number of good rows G follows the binomial distribution with parameters M and π. Thus, by using the union bound over all possible choices of i (and we have n of them), we obtain $\Pr\{\mathrm{E}_1\} \leq n \Pr\{G = 0\}$, and hence

$$\Pr\{\mathrm{E}_1\} \leq n(1 - \pi)^M. \tag{13}$$

We now analyze the probability of E_2, which is referred to as $\Pr\{\mathrm{E}_2\}$. Since $\mathbf{y}_1$ and $\mathbf{y}_2$ are distinct, then there exists an index $i^\star$ such that the $i^\star$-th component of $\mathbf{y}_1$ is 1 and the $i^\star$-th component of $\mathbf{y}_2$ is zero, or vice versa. According to the Z-channel in Fig. 2 the 1 can be flipped into a 0 with probability $1/q$, whereas the 0 remains 0. Thus, we have

$$\Pr\{\mathrm{E}_2\} = 1/q, \tag{14}$$

which vanishes as the field size q increases.

The probability of error $\Pr\{\mathrm{E}\}$ is hence given by $\Pr\{\mathrm{E}_1 \cup \mathrm{E}_2\}$ which, by using the union bound can be upper bounded as $\Pr\{\mathrm{E}\} \leq \Pr\{\mathrm{E}_1\} + \Pr\{\mathrm{E}_2\}$. Moreover, for given values of n and ℓ, the value of π in (13) can be chosen to be as large as possible. This concludes the proof of Theorem 3.

REFERENCES

[1] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv:1806.00939*, 2018.

[2] "IBM World Community Grid," https://www.worldcommunitygrid.org.

[3] "Folding@Home," https://foldingathome.org.

[4] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, vol. 70, 2017, pp. 3368–3376.

[5] Q. Yu, M. A. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Inf. Processing Systems*, 2017, pp. 4406–4416.

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[7] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[8] S. Dutta, V. Cadambe, and P. Grover, ""short-dot": Computing large linear transforms distributedly using coded short dot products," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, 2016, pp. 2100–2108.

[9] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure coded computing using secret sharing via staircase codes," *CoRR*, vol. abs/1802.02640, 2018.

[10] R. Dorfman, "The detection of defective members of large populations," *Ann. Math. Statist.*, vol. 14, no. 4, pp. 436–440, 12 1943.

[11] D.-Z. Du and H. FK, *Combinatorial Group Testing And Its Applications*, 2000, vol. 2nd ed. World Scientific Publishing Company.

[12] E. A. Rawashdeh, "A simple method for finding the inverse matrix of Vandermonde matrix," *Matematički Vesnik*, 2018.

[13] M. Cheraghchi, A. Hormati, A. Karbasi, and M. Vetterli, "Group testing with probabilistic tests: Theory, design and application," *IEEE Trans. on Inf. Theory*, vol. 57, no. 10, pp. 7057–7067, 2011.